Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A processor comprising:

an out-of-order microinstruction pointer (μIP) stack for storing pointers in a microcode (μcode) execution core.

2. (Original) The processor of claim 1 in which the μIP stack comprises:

an entry number field;

a microinstruction pointer (μIP) field;

a back pointer field;

a retirement indicator field; and

a return pointer field.

3. (Original) The processor of claim 2 in which the μIP field is 14-bits wide.

4. (Original) The processor of claim 3 in which the μIP field has a microinstruction pointer (μIP) pushed by a first microoperation (μOp) code and used by a second μOp code.

5. (Original) The processor of claim 2 in which the back pointer field has a pointer to a next entry in the μIP stack for a micro-type of service (μTOS) bit to point to after a μOp.

6. (Original) The processor of claim 2 in which the retirement indicator field has an indication of whether an entry has retired.

7. (Original) The processor of claim 2 in the return pointer field a pointer to a location in a retirement stack to which an entry is copied after being retired.

8. (Currently amended) A method executed in a processor comprising:

executing microcode (μcode) ~~stored~~ addressed by pointers stored in an out-of-order microinstruction pointer (μIP) stack; and

manipulating the μIP stack with a set of microinstructions.

9. (Original) The method of claim 8 in which the stack has an entry number field, a microinstruction pointer (μIP) field, a back pointer field, a retirement indicator field and a return pointer field.

10. (Original) The method of claim 9 in which the μIP pointer field is 14-bits wide.

11. (Original) The method of claim 10 in which the μIP pointer field has a microinstruction pointer (μIP) pushed by a first microoperation (μOp) code and used by a second μOp code.

12. (Original) The method of claim 9 in which the back pointer field has a pointer to a next entry in the μIP stack for a micro-type of service (μTOS) bit to point to after a μOp.

13. (Original) The method of claim 9 in which the retirement indicator field has an indication of whether an entry has retired.

14. (Original) The method of claim 9 in which the return pointer field contains a pointer to a location in a retirement stack to which an entry is copied after being retired.

15. (Original) The method of claim 9 in which manipulating comprises:

pushing a next μIP on to the μIP stack; and

using the next μIP in an intermediate field as a target μIP in a jump operation.

16. (Original) The method of claim 9 in which manipulating comprises:

taking a value of an intermediate field of a microoperation (μOp); and

pushing the value on to the μIP stack.

17. (Original) The method of claim 9 in which manipulating comprises:

popping a value off the μIP stack; and

replacing a current μOp intermediate field.

18. (Original) The method of claim 9 in which manipulating comprises:

popping a value off of the μIP stack; and

jumping to that value.

19. (Original) The method of claim 9 in which manipulating comprises:

reading a value off the μIP stack; and

replacing a μOp's intermediate field with the value.

20. (Original) The method of claim 9 in which manipulating comprises setting the μIP stack
pointers to reset.

21. (Original) The method of claim 9 further comprising providing a set of pointers that point to
different entries in the μIP stack.

22. (Original) The method of claim 21 in which the set of pointers includes a μTOS pointer that
points to a top of the μIP stack.

23. (Original) The method of claim 21 in which the set of pointers includes a µAlloc pointer that points to a next allocated entry in the µIP stack.

24. (Original) The method of claim 21 in which the set of pointers includes a NextRet pointer that points to a next entry in the µIP stack to be deallocated.

25. (Original) The method of claim 21 in which the set of pointers includes µRetTos pointer that points at a retired top of the µIP stack.

26. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_call µOP that takes a next µIP, pushes the next µIP on the µIP stack, and uses the next µIP in an intermediate field as a target µIP of a jump.

27. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_push µOP that takes a value in an intermediate field and pushes the value on the µIP stack.

28. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_pop µOP that pops a value off the µIP stack and replaces the value with the µOP's intermediate field.

29. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_return µOP that pops a value off of the µIP stack and jumps to that µIP.

30. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_tos_read µOP that reads a value off the µIP stack and replaces this µOP's intermediate field.

31. (Currently amended) The method of claim 8 in which the a plurality of µOPs include an ms_µip_stack_clear µOP that sets the µIP stack pointers to reset.

32. (Currently amended) A computer program product residing on a computer readable medium having instructions stored thereon which, when executed by the processor, cause the processor to:

 execute microcode (μcode) ~~stored~~ <u>addressed through pointers stored</u> in an out-of-order microinstruction pointer (μIP) stack; and

 manipulate the μIP stack with a set of microinstructions.

33. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

 push a next μIP on to the μIP stack; and

 use the next μIP in an intermediate field as a target μIP in a jump operation.

34. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

 take a value of an intermediate field of a microoperation (μOp); and

 push the value on to the μIP stack.

35. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

 pop a value off the μIP stack; and

 replace a current μOp intermediate field with the value.

36. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

 pop a value off of the μIP stack; and

 jump to that value.

37. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

read a value off the µIP stack; and

replace a µOp's intermediate field with the value.

38. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

set the µIP stack pointers to reset.